47

Notice of Allowability	**Application No.** 09/930,541	**Applicant(s)** ARBOUZOV ET AL.
	Examiner Todd Ingberg	**Art Unit** 2193

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to _12/23/2005_.
2. ☒ The allowed claim(s) is/are _1-20_.
3. ☒ The drawings filed on _14 August 2001_ are accepted by the Examiner.
4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.
6. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
   1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).
7. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☒ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO-1449 or PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application (PTO-152)
6. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____ .
7. ☒ Examiner's Amendment/Comment
8. ☒ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

EXAMINER'S AMENDMENT

1. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

The claims of the application has been amended as follows:

The Examiner has capitalized the trademark JAVA as required by the MPEP.

Marked Copy

Claim 1
A method for use in a computing system for preprocessing [Java] JAVA application code, comprising the operations of:

receiving a [Java] JAVA template file, the [Java] JAVA template file including [Java] JAVA programming language code and meta code. The meta code being a computer program written in meta language using a programming language;

processing the [Java] JAVA template to create an intermediate program using the meta code, wherein the intermediate program is a [Java] JAVA program;

compiling the intermediate program to create an intermediate, class, wherein the intermediate class is a [Java] JAVA based class; and

generating an object text file using the intermediate class.

Claim 2
A method as recited in claim 3, wherein the meta code can include [Java] JAVA programming language statements.

Claim 3
A method as recited in claim 1, wherein the meta code is equivalent to the [Java] JAVA programming language.

Claim 4
A method as recited in claim 1, further comprising the operation of transforming lines in the [Java] JAVA template.

Claim 8
A method for use in a computer system for transforming lines of a [Java] JAVA template for preprocessing, comprising the operations of:

obtaining a line of text from a [Java] JAVA template;

determining if the line of text begins with a meta symbol, wherein the meta symbol is a predefined symbol indicating the line of text is a preprocessor directive;

removing the meta symbol from the line of text when the line of text begins with the meta symbol; and

transforming the line of text into a function argument when the line of text does not begin with the meta symbol.

Claim 16
A [Java]_JAVA preprocessor for use in a computing system, comprising:

a meta code converter capable of processing a [Java]_JAVA template to create an intermediate program using meta code included in the [Java]_JAVA template; and

an object text generator that compiles the intermediate program to create an intermediate class, wherein the intermediate class is a [Java]_JAVA based class, the object text generator also capable of generating an object text filed using the intermediate class.

Claim 18
A [Java]_JAVA preprocessor as recited in claim 17, wherein the meta language uses the [Java]_JAVA programming language.

Claim 19
A [Java]_JAVA preprocess :as recited in claim 18, further comprising a preprocessor library interface module that specifies a contract that can be extended by any [Java]_JAVA preprocessor library that is to be supported by the [Java] JAVA preprocessor.

Unmarked Copy

Claim 1
A method for use in a computing system for preprocessing JAVA application code, comprising the operations of:

receiving a JAVA template file, the JAVA template file including JAVA programming language code and meta code. The meta code being a computer program written in meta language using a programming language;

processing the JAVA template to create an intermediate program using the meta code, wherein the intermediate program is a JAVA program;

compiling the intermediate program to create an intermediate, class, wherein the intermediate class is a JAVA based class; and

generating an object text file using the intermediate class.

Claim 2
A method as recited in claim 3, wherein the meta code can include JAVA programming language statements.

Claim 3

A method as recited in claim 1, wherein the meta code is equivalent to the JAVA programming language.

Claim 4
A method as recited in claim 1, further comprising the operation of transforming lines in the JAVA template.

Claim 8
A method for use in a computer system for transforming lines of a JAVA template for preprocessing, comprising the operations of:

obtaining a line of text from a JAVA template;

determining if the line of text begins with a meta symbol, wherein the meta symbol is a predefined symbol indicating the line of text is a preprocessor directive;

removing the meta symbol from the line of text when the line of text begins with the meta symbol; and

transforming the line of text into a function argument when the line of text does not begin with the meta symbol.

Claim 16
A JAVA preprocessor for use in a computing system, comprising:

a meta code converter capable of processing a JAVA template to create an intermediate program using meta code included in the JAVA template; and

an object text generator that compiles the intermediate program to create an intermediate class, wherein the intermediate class is a JAVA based class, the object text generator also capable of generating an object text filed using the intermediate class.

Claim 18
A JAVA preprocessor as recited in claim 17, wherein the meta language uses the JAVA programming language.

Claim 19
A JAVA preprocess :as recited in claim 18, further comprising a preprocessor library interface module that specifies a contract that can be extended by any JAVA preprocessor library that is to be supported by the JAVA preprocessor.

REASONS FOR ALLOWANCE

I. FORMAL MATERS

Drawings

2. The correction to drawing was received on December 23, 2004 and have been approved.

Claim Rejections - 35 USC § 101

3. Rejection under 35 U.S.C. 101 has been overcome by amendment.

Knowledge of the Ordinary Artisan

4. The following information should be known to one of ordinary skill in the art at the time of invention.

The basic workings of the C++ preprocessor - The C++ preprocessor reads the file and locates the directives in the C++ file. Directives begin the line with a "#". The following are C++ directives and are located on pages 424 to 425 of the Design and Evolution of C++ book written by the inventor of C++, Bjarne Stroustrup. This book is a historical perspective of design decisions he made in the development of the programming language.

include

- Make interface definition available.
- Compose source text

#define

- Define symbolic constants
- Define open subroutines
- Define generic subroutines
- Define generic "types"
- Renaming
- String concatenation

- Define special purpose syntax
- General macro processing

#ifdef

- version control
- Commenting out code

#pragma

- Control of layout
- Informing the compiler about unusual control flow

The result of the preprocessor is a file ready for compilation by the C++ compiler. This does not guarantee the code included or generates will pass the syntax of the C++ compiler. The syntax is dependent on the correctness of the code written to include/expand. The directive "#" is removed because it's purpose is for the preprocessor to identify the present of a directive, the compiler does not recognize the directives and will provide a syntax error.

Terms Interpretations

5. The following meaning of the terms. The Examiner scanned the Applicant's clarification into the record. If a scanning error exists it is not intended to be an alteration of Applicant's statements.

A. Meta language:

The Office has interpreted that the meta language refers to the JAVA programming language. The Applicants respectfully disagree with the Office's interpretation. One c ordinary skill in the art knows that meta languages are used to describe another language (e.g here used in preprocessor). See Alan Freedman, The Computer Desktop Encyclopedia, 56 (2nd ed. 1999). While in the subject application the meta language can use the Java programming language, the meta language, as described in the subject application, should not be interpreted to be equivalent and/or limited to the JAVA programming language. By way c example, the meta language of the claimed invention can be C++, C, or any other selected language. As such, the Office's interpretation of the meta language is not the same as the Applicants' interpretation.

B. Header file:

This term has been interpreted by the Office to be "a file containing a PROLOG (i.e., a documents' area where programmers document the programs' authors and history), and in short a file containing comments. The Applicants respectfully submit that the Office's interpretation of the header file is not the same as the Applicants. First, in the specification of the subject application, header code is used and not header file. See page 10, lines 16-23. Second, the header code, referred to as the "prolog" in the claimed invention, can include some copyright information, JAVA language `package' and `import' clauses, and intermediate program class declaration with an opening brace `(`. Such definition, however, is not equivalent to "a file containing comments."

C. Footer file:

This term has been interpreted by the Office to be a file containing source code such as class files (e.g., the header file in C++). The Applicants respectfully submit that the Office's interpretation of the footer file is not the same as the Applicants'. Again, in the specification of the subject application, footer code or footer data is used and not footer file. See page 10, lines 16-23 and page 11, lines 21-23. Furthermore, the footer code or data is not the same as the header file in C++. Rather, as provided on page 18, lines 12-13 of the subject application, the footer data can include the definition of the `main' method and a closing brace `) '.

D. Intermediate program:

This term is interpreted to be the output from the preprocessor ready to be used as input to the JAVA compiler. The Applicants respectfully submit that the Office's interpretation of the term intermediate program is not the same as the Applicants'. In the subject application, the JAVA object text file is to be compiled by the compiler so as to generate a JAVA byte-codes file. See page 10, lines 5-6. In the subject application, contrary to the Office's interpretation, the intermediate program is generated as a result of the JAVA object text file being processed by the first preprocessing stage. Thereafter, the "intermediate program" is compiled by the JAVA compiler so as to generate the executable intermediate program. At this point, the executable intermediate program is executed generating the "intermediate source." Then, the intermediate source is compiled by the JAVA compiler so as to generate the executable code. In the prior art preprocessors such as C/C++, however, the source. file is preprocessed directly generating the "intermediate source" which is then compiled by the C/C++ compiler so as to generate the executable code. Accordingly, while the C/C++ compiler compiles the "intermediate source" so as to arrive at the executable code, in the subject application, the executable intermediate program of the subject application is executed so as to create the intermediate source.

F. Meta code:

This term has been interpreted by the Office to be the preprocessor file that is an input to the preprocessor that includes directives that may or may not have arguments and JAVA code. The Applicants respectfully disagree with the Office's interpretation. On page 2, lines 13-14, the Applicants state that meta code is lines of text in a source file, which start with -a particular symbol. On page 8, lines 4-6, the Applicants state that the JAVA template text file comprises a JAVA language program and meta code for use with the JAVA preprocessor. Thus, contrary to the Office's interpretation, meta code cannot constitute the entire preprocessor file input. Rather, the preprocessor file input includes meta code (i.e., lines of meta code) as well as the source code.

Additionally, on page 12, lines 2-8, the Applicants provide:

In addition to processing normal object text, the JAVA preprocessor of the present invention if further capable of processing string mode text. String mode text is text that does not begin with a meta symbol and occupies multiple lines in a file. More particularly, the line of text includes LineFeed or end-of-line symbols. As described in greater detail subsequently, the preprocessor of the embodiments of the present invention is capable of operating on strings that occupy multiple lines as though the string comprised a single line of text.

Thus, contrary to the Office's interpretation, the meta code of the claimed invention can be any construct of JAVA, and may or may not include a directive with or without arguments.

G. Meta symbol:

This term has been interpreted to be the actual indicator that a line is meant to be processed by the preprocessor (i.e., formally called. a directive.). The Applicants herein submit that a line of the meta code can be required to be processed by the preprocessor even if the line of meta code does not include a directive.

II. REASONS FOR ALLOWANCE

6. The following is an examiner's statement of reasons for allowance:

The limitations of claim 1 represent the allowable limitations of the independent claims.

Claim 1

A method for use in a computing system for preprocessing JAVA application code, comprising the operations of:

receiving a JAVA template file, the JAVA template file including JAVA programming language code and meta code. The meta code being a computer program written in meta language using a programming language;

processing the JAVA template to create an intermediate program using the meta code, wherein the intermediate program is a JAVA program;

compiling the intermediate program to create an intermediate, class, wherein the intermediate class is a JAVA based class; and

generating an object text file using the intermediate class.

<u>Attack on Motivation</u>
The Applicant's wrote:

"For at least the reasons provided below, none of the combinations of the cited prior art raise a prima facie case of obviousness against the subject matter defined in independent claims 1, 8, and 16 because as a whole and in its entirety, the JAVA reference leads away from using a preprocessor in the JAVA language. It is respectfully submitted that the Office has dismissed the portions of the JAVA Reference leading away from implementing a preprocessor with the JAVA language. The Office's attention is drawn to the following excerpts:

JAVA has also done away with the standard #define macros. Actually, they've done away with the C preprocessor altogether. This is continuing with the "enforced OOP ideology. JAVA doesn't use header files #defines when compiling source code. Say good-bye to those huge header files listing function prototypes, typedefs, defines. etc.".

This attack on motivation is not persuasive. Prior art search revealed related work in the area of adding a pre-processor to the JAVA programming language. Many reference on JAVA explicitly mention JAVA is attempting to look like C++. The claimed invention supports the inputting of a JAVA template file with meta data. A working definition of meta data is "data about data". Examples of meta data in Object Technology is the relationships among object "part-of", "has-a", cardinality etc. On FAOM the Examiner had mistakenly placed the interpretation of meta data behind the preprocessor. The JAVA programming languages ability to support modeling of real world problems with relationships in JAVA code. This information is the by product of modeling weather the model is a the generation of a program which models a real world problem or is the output of a visual model. The claimed invention supports the later. The template file contains meta data which describes the relationships among objects. This meta data is input to the instant invention the JAVA preprocessor. The result is the generation of an intermediate program.

The closest prior art of record is Rational Rose/C++ Chapter 3 Reverse Engineering pages 131 to 185. Reference when taken with the prior art of record does not explicitly teach the intermediate steps of the claimed invention in combination of the claim limitations. the lack of detail does not support a rejection under the Zurko vs. Dickenson where Official Notice and arguments based on Basic knowledge or common sense which are not supported by evidence in the record ***lack*** substantial evidence support.

The following are Applicant's arguments that are persuasive.

"Furthermore, the Applicants respectfully submit that all the combined elements and limitations of independent claims 1, 8, and 16 have not been considered as a whole by the Office. For instance, the Office has not demonstrated a prior art teaching or suggestion of the intermediate program, compiling of the intermediate program to create and intermediate class, and generating of an object text file from the ;intermediate class (as defined in claim 1); transforming of a line text into a function argument when a line of text in the source code does not begin with a meta symbol (as defined in independent claim 8); and a meta code converter capable of converting a JAVA template to create an intermediate program, and an object generator that compiles the intermediate program to create the intermediate class, and thereafter an object text file (as defined in independent claim 16).

. . .

"Yet further, the combinations of the cited references fail to disclose, teach, or suggest all the claim limitations as provided in the claimed invention. By way of example, among other features, the JAVA preprocessor has the capability to preprocess both, statements, which have directives, and statements, which do not have directives. This is contrary to the teachings of the C/C++ preprocessors wherein preprocessing is limited to only statements having directives. In fact, even if the JAVA Reference and the C++ Reference were combinable and were to be combined (a preposition with which the Applicants disagree), the combination of the two references would not have disclosed, suggested, or taught preprocessing statements not have directives.

Still further, in the claimed invention, the meta language can use an entire programming language, e.g., JAVA. In the C++ or C preprocessors, the meta language has not been taught, disclosed, or suggested to be capable of using an entire programming language. Furthermore, in the claimed invention, the meta language can use any programming language, and is not limited to the JAVA programming language, only. The C or C++ preprocessor, however, do not have such capabilities."

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Correspondence Information

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to **Todd Ingberg** whose telephone number is (703) 305-9775. The examiner can normally be reached during the following hours:

Monday	Tuesday	Wednesday	Thursday	Friday
6:15 – 1:30	6:15- 3:45	6:15 – 4:45	6:15-3:45	6:15-130

This schedule began December 1, 2003 and is subject to change.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, **Kakali Chaki** can be reached on (703) 305-9662. Please, note that as of August 4, 2003 the **FAX number** changed for the organization where this application or proceeding is assigned is **(703) 872-9306**.

Also, be advised the United States Patent Office **new address** is

Post Office Box 1450

Alexandria, Virginia 22313-1450

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-9700.

[illegible]

Todd Ingberg
Primary Examiner
Art Unit 2124
June 13, 2005